# Programação Visual
## Trabalho de Laboratório nº 1

| | |
|---|---|
| **Objetivo** | Programação em C# - sintaxe avançado.<br>Criação de aplicações de consola simples. |
| **Programa** | Protótipo da aplicação Jogos de Cartas – *refactoring* para C# da aplicação criada em C# usando as simplificações introduzidas na sintaxe do C#. |
| **Regras** | Use as convenções de codificação adotadas para a linguagem C#.<br>Na classe do programa não coloque atributos nem crie nenhum método para além do **`Main`**.<br>Não é necessário obter dados do utilizador. Forneça os dados ao nível do código. |

**Descrição**

**Nível 1**

- Descarregue e descompacte a aplicação Java **CardGames** fornecida com este enunciado. Pretende-se fazer o *refactoring* desta aplicação para C# utilizando algumas das características da sintaxe avançada do C#
- Comece por criar o tipo enumerado **`Suit`** que representa o naipe de cartas. Use os mesmos valores que foram definidos na aplicação em Java.
  **Notas:** Em C# não é possível incluir métodos nos tipos enumerados, por isso **não defina** o método **`toString()`** neste tipo **`Suit`**. Tenha ainda em atenção a convenção do C# para os valores dos tipos enumeradas que devem usar **PascalCase**.
- Tendo em conta as notas anteriores crie agora o tipo enumerado **`FaceName`**.
- Crie agora a classe **`Card`**. Use propriedades implícitas para o naipe e para o valor. Inclua os construtores definidos na classe equivalente em Java e acrescente o construtor sem argumentos que inicializa o valor com **`-1`** e o naipe com **`None`**. Ainda, em vez do equivalente ao método **`getName`** inclua na classe uma propriedade abstrata **`Name`** apenas de leitura. **Nota:** por enquanto não implemente na classe **`Card`** o método equivalente ao **`compareTo`** do Java e à interface associada.

**Nível 2**

- Crie a classe **`NumberedCard`**. Inclua os construtores da classe equivalente em Java e acrescente o construtor sem argumentos. Defina também o atributo **`number`** e uma propriedade **`Number`** que devolve o valor deste atributo. Esta propriedade deve ser apenas de leitura e usar a notação *expression body*. Para a implementação do método **`getName`** use a redefinição da propriedade **`Name`** definida na classe **`Card`**. Esta propriedade é apenas de leitura. Defina ainda o método **`ToString()`**.
- Crie a classe **`FaceCard`** de uma forma semelhante à classe **`NumberedCard`**, incluindo as propriedades **`Name`** e **`FaceName`** apenas de leitura, com notação *expressionBody*, e os construtores existentes. Acrescente também o construtor sem argumentos e a redefinição do método **`ToString()`**.
- Teste o código criado reproduzindo no método **`Main`** a criação da lista de cartas e a visualização das mesmas como está no método equivalente da aplicação em Java. Para a criação das diferentes cartas use a sintaxe de inicialização de objetos do C#.

# Programação Visual
## Trabalho de Laboratório nº 1

**Nível 3**

- Pretende-se agora resolver o problema da implementação dos métodos **`ToString`** nos tipos enumerados **`Suit`** e **`FaceName`**. Uma vez que o C# não permite a criação de métodos nos tipos enumerados vai-se resolver o problema definindo métodos de extensão para estes tipos. Sendo assim, crie uma classe estática **`CardUtils`** e defina nesta classe dois métodos de extensão com o nome **`ToNameString()`** aplicados aos tipos **`FaceName`** e **`Suit`**. Estes métodos devem retornar uma **`string`** com o texto do valor de acordo com o que existe nos tipos equivalentes da aplicação em Java.
- Adapte onde for necessário, as classes das cartas para que utilizem o método **`ToNameString()`** criado em vez do **`ToString()`** quando devolverem o texto desses tipos enumerados.
- Verifique que a execução do programa já está corrigida com o texto das peças a aparecer em português.

**Nível 4**

- Crie a classe **`Deck`** de acordo com a classe equivalente da aplicação em Java. Nesta classe, guarde a coleção de cartas num objeto da classe genérica **`List<>`** (como o **`ArrayList<>()`** do Java). Neste caso, não será possível usar a classe **`Stack<>`** do C# porque esta não tem as operações de inserção e remoção de um elemento numa determinada posição da coleção requeridas em alguns dos métodos implementados. Implemente todos os métodos da classe da aplicação em Java à exceção dos métodos **`sortByValue`**, **`shuffle`**, **`sort`** e **`iterator`**.
  Notas: No método **`ToString`** use a classe **`StringBuilder`** para concatenar as várias *strings*. Use uma propriedade de leitura **`Cards`** em vez do método **`GetCards`**.
- Teste reproduzindo o código respetivo da aplicação em Java.

**Nível 5**

- Crie a classe **`SuecaDeck`** à semelhança da classe equivalente da aplicação Java. Para obter os valores do tipo enumerado **`Suit`** use: **`Enum.GetValues(typeof(Suit))`** .
- Teste reproduzindo o código respetivo da aplicação em Java.
- Pretende-se agora implementar os métodos **`sortByValue`** e **`shuffle`** da classe **`Deck`**. Sendo assim, comece por implementar a interface **`IComparable<Card>`** na classe **`Card.`** Esta interface é semelhante à interface **`Comparable<>`** do Java. Esta interface é usada pelo método **`Sort`** da coleção **`List<Card>`**. Utilize este método na implementação do novo método **`SortByValue`** da classe **`Deck`**. Implemente também o método **`Shuffle`** desta classe. Terá de definir um algoritmo para baralhar as cartas uma vez que este método não existe nas coleções do C#.
- Volte a testar acrescentando, mais uma vez, o código respetivo da aplicação em Java.

**Desafio**

- Complete o desenvolvimento da aplicação criando os métodos equivalentes dos métodos **`Iterator`** e **`Sort`** da aplicação Java que faltam na classe **`Deck`** e implemente também o código de teste do programa principal.

**Notas**

Para os identificadores siga as convenções adotadas pelo C#, nomeadamente:

- A notação camelCase para o nome das variáveis locais e identificadores privados.
- A notação PascalCase para os nomes públicos dos métodos, classes e interfaces.
- Não utilize o símbolo '_' nos identificadores nem abreviaturas